08｜变换量化：如何减少视觉冗余？

2021-12-08 李江

《攻克视频技术》 课程介绍 >

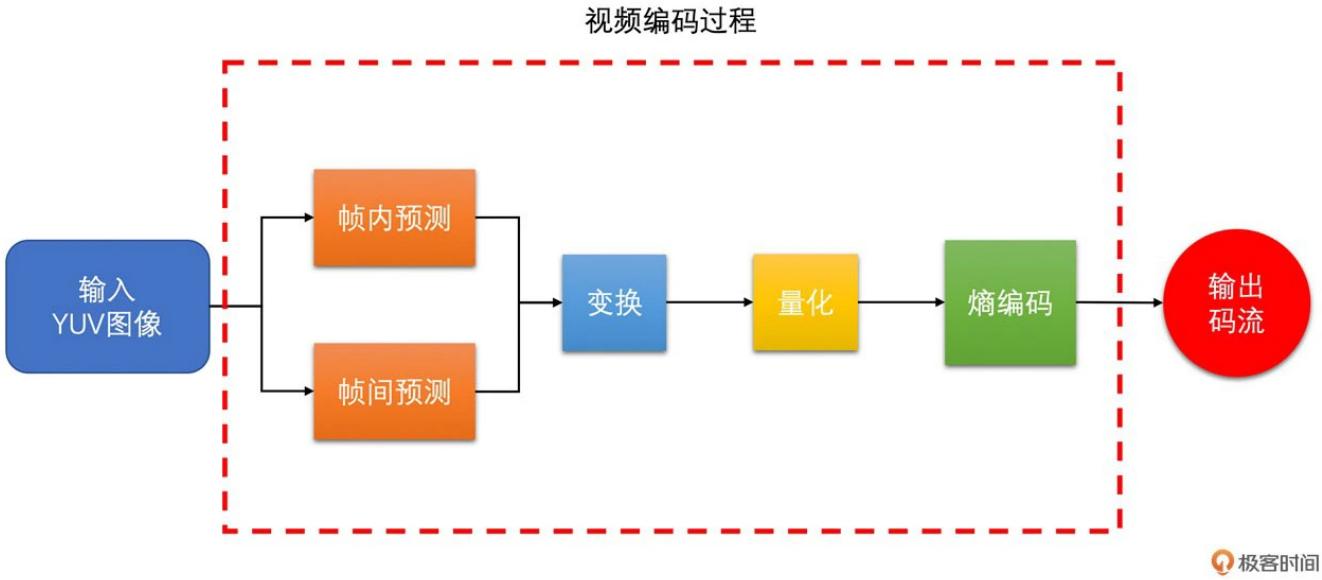
讲述：李江

时长 13:34 大小 12.43M

你好，我是李江。

在前面几节课里面我们对视频编码的原理、帧内编码和帧间编码都做了详细的介绍。我们知道，通过帧内编码可以去除空间冗余，通过帧间编码可以去除时间冗余，而为了分离图像块的高频和低频信息从而去除视觉冗余，我们需要做 DCT 变换和量化。

为了让你更透彻地了解视频编码中的 DCT 变换和量化的原理，在这节课里面，我们会对 DCT 变换和量化的过程做一个深入的探讨。通过下图你可以很清楚地看到视频编码的过程，并且能够更直观地感受 DCT 变换和量化在整个视频编码过程中的重要性。



由于 DCT 变换和量化过程是一个跟数学比较相关的过程，且大多数是数学计算。因此今天的课程中数学公式和计算过程相对会多一些。但是总体来说难度不是很大。

下面我们就先来讨论一下常规的视频编码中的 DCT 变换和量化，看看它是怎么去除一部分高频信息来达到去除视觉冗余的。并且，由于 H264 中用到的 DCT 变换和量化跟常规的 DCT 变换和量化有一些区别，其主要在于 H264 使用整数变换代替常规的 DCT 变换，并将 DCT 变换中的一部分计算整合到量化中，从而减少浮点运算漂移问题。因此，我们还会对 H264 中的 DCT 变换和量化做一下介绍，最后对比一下 H264 中的变换和量化与常规的变换和量化的区别。

常规视频编码中的 DCT 变换和量化

通过前面的学习，我们知道 DCT 变换和量化的目的是去除视觉冗余。接下来让我们看一下常规视频编码中 DCT 变换和量化是如何实现这一目的的。

DCT 变换

DCT 变换，就是离散余弦变换。它能够将空域的信号（对于图像来说，空域就是你平时看到的图像）转换到频域（对于图像来说，就是将图像做完 DCT 变换之后的数据）上表示，并能够比较好的去除相关性。其主要用于视频压缩领域。现在常用的视频压缩算法中基本上都有 DCT 变换。

在视频编码原理那节课中我们也讲过，图片经过 DCT 变换之后，低频信息集中在左上角，而高频信息则分散在其它的位置。通常情况下，图片的高频信息多但是幅值比较小。高频

信息主要描述图片的边缘信息。

由于人眼的视觉敏感度是有限的，有的时候我们去除了一部分高频信息之后，人眼看上去感觉区别并不大。因此，我们可以先将图片 DCT 变换到频域，然后再去除一些高频信息。这样我们就可以减少信息量，从而达到压缩的目的。

DCT 变换本身是无损的，同时也是可逆的。我们可以通过 DCT 变换将图片从空域转换到频域，也可以通过 DCT 反变换将图片从频域转回到空域。下面我们来看一下 DCT 变换的公式。

一维 DCT 变换公式如下，其中 f(i) 是指第 i 个样点的信号值，N 代表信号样点的总个数。

$$F(u) = c(u)\sum_{i=0}^{N-1} f(i)\cos\left[\frac{(i+0.5)\pi}{N}u\right]$$

$$\text{其中：} c(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & \text{其他} \end{cases}$$

极客时间

二维 DCT 变换公式如下，其中 f(i, j) 是指第 (i, j) 位置的样点的信号值，N 代表信号样点的总个数。

$$F(u,v) = c(u)c(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\cos\left[\frac{(i+0.5)\pi}{N}u\right]\cos\left[\frac{(j+0.5)\pi}{N}v\right]$$

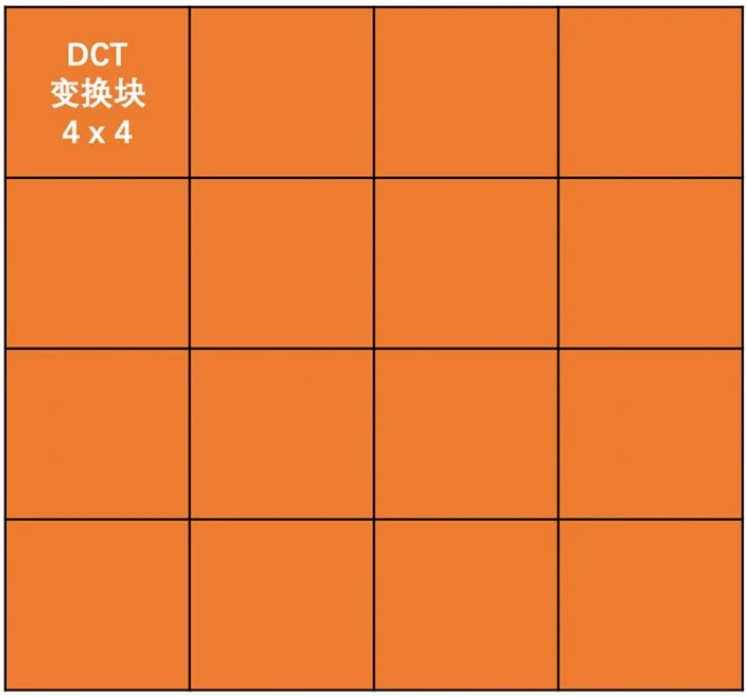
$$\text{其中：} c(u)\text{和}c(v) = \begin{cases} \sqrt{\frac{1}{N}}, & u、v = 0 \\ \sqrt{\frac{2}{N}}, & \text{其他} \end{cases}$$

极客时间

一般在编码标准中图像是进行二维 DCT 变换的，因为图像是个二维信号。但是实际上在代码里面我们经常将二维 DCT 变换转换成两个一维 DCT 变换来进行。

在视频压缩中，DCT 变换是在帧内预测和帧间预测之后进行的。也就是说，DCT 变换其实是对残差块做的。我们在编码时会将图像划分成一个个宏块，而宏块又可以划分成一个个子块。那 DCT 变换是在宏块上进行还是在子块上进行呢？

其实，**通常情况下 DCT 变换是在 4x4 的子块上进行的**（也可以在 8x8 子块上进行，但是只有在扩展 profile 才支持，由于原理是一样的，因此这里不再展开讨论），即便预测时并没有对宏块再做划分。也就是说，不管宏块有没有被划分到 4x4 的子块，我们在做 DCT 变换时，都是在一个个 4x4 块上进行的。如下图所示：



极客时间

好了，如果我们将上面的 DCT 变换公式用在 4x4 的变换块上，则 4x4 的 DCT 变换就可以通过下面的 4x4 的矩阵乘法来表示了。

$$Y = AXA^T$$

其中：
Y是变换后的4x4变换块
X是4x4残差块

$$A = \begin{bmatrix} \frac{1}{2}cos(0) & \frac{1}{2}cos(0) & \frac{1}{2}cos(0) & \frac{1}{2}cos(0) \\ \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{5\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{7\pi}{8}) \\ \sqrt{\frac{1}{2}}cos(\frac{2\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{6\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{10\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{14\pi}{8}) \\ \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{9\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{15\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{21\pi}{8}) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) \end{bmatrix}$$

极客时间

为了让你更好地理解 DCT 变换，我们通过下面的例子来看一下 4x4 的残差块的 DCT 变换结果。我们称左上角的系数为 DC 系数，而其它系数为 AC 系数。

9	2	−5	−5
14	−1	−11	−7
15	−3	−18	−9
16	−3	−22	−11

4x4残差块

极客时间

对这个 4x4 的残差块运用公式，可得：

$$Y = AXA^T$$

其中:
Y是变换后的4x4变换块
X是4x4残差块

$$Y = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & -\sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) \end{bmatrix} \begin{bmatrix} 9 & 2 & -5 & -5 \\ 14 & -1 & -11 & -7 \\ 15 & -3 & -18 & -9 \\ 16 & -3 & -22 & -11 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & \frac{1}{2} & \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) \\ \frac{1}{2} & \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & -\frac{1}{2} & -\sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & -\frac{1}{2} & \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & \frac{1}{2} & -\sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) \end{bmatrix}$$

极客时间

我们可以得到下面的图：

9	2	-5	-5
14	-1	-11	-7
15	-3	-18	-9
16	-3	-22	-11

4x4残差块

-10	35	21	-5
8	-9	-9	3
0	-1	-2	0
0	-1	0	0

DCT变换后的块

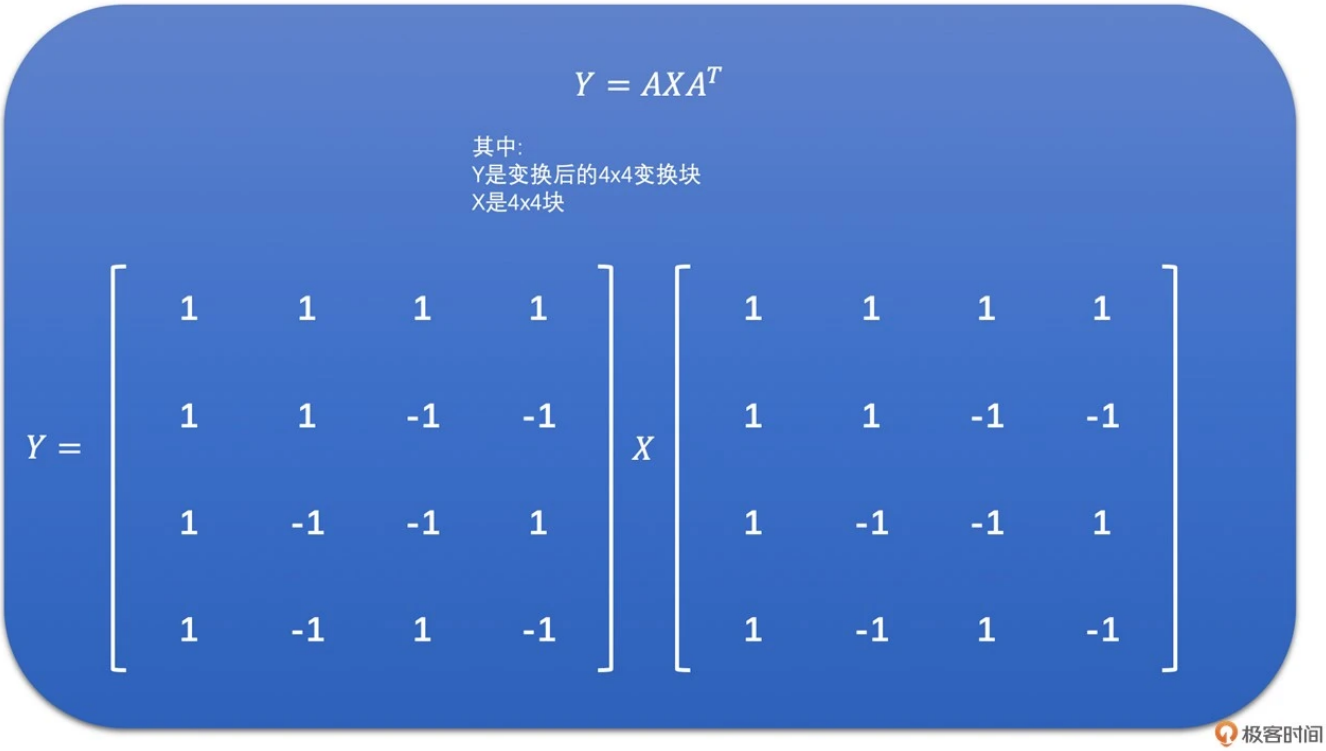
极客时间

从上面 DCT 变换的公式和矩阵表示方式中，我们可以看到，DCT 变换的计算过程中涉及到了 cos 函数。那也就是说计算的过程中一定涉及到了浮点运算。而浮点运算计算速度比较慢。**那有没有什么运算可以将图像块比较快速的转换到频域呢？**答案肯定是有的。前面我们其实已经讲到过，那就是 Hadamard 变换，也叫哈达玛变换。

Hadamard 变换

在视频编码过程中，Hadamard 变换也经常会用到。前面我们在帧内预测的率失真优化的模式选择里就讲到过，Hadamard 变换可以代替 DCT 变换将残差块快速转换到频域，以便用来估计一下当前块编码之后的大小。

其实在 H264 的亮度 16x16 帧内预测块和色度 8x8 预测块中也会使用到 Hadamard 变换。稍后我们会在 H264 中的 DCT 变换和量化部分对它进行简单介绍。下面我们先来看一下 Hadamard 变换的矩阵表示形式。



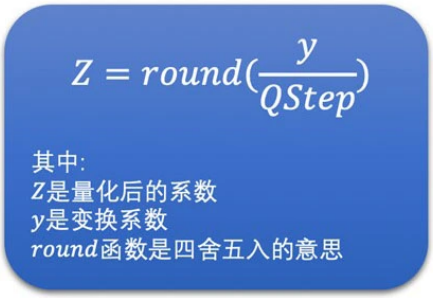
你是不是可以看到，Hadamard 变换是没有浮点运算的？因此其计算速度很快，并且也能够将图像块从空域变换到频域。因此，我们可以用它一定程度上粗略的代替 DCT 变换，从而用来简化运算。

好了，现在我们知道了如何通过 DCT 变换和 Hadamard 变换将残差块从空域转换到频域。接下来我们看看在常规视频编码中量化是如何做到通过去除一部分高频信息来最终达到去除视觉冗余的。

量化

前面我们讲了，我们将图像块变换到频域之后，AC 系数比较多，但是一般幅值比较小。并且，我们可以去除一些 AC 系数，达到压缩图像的目的，同时人眼看起来差距不大。**这个去除 AC 系数的操作是什么呢？**很明显就是量化了。

其实**量化的操作并不是针对 AC 系数去做的，DC 系数也同样会做量化**，只是通常情况下，DC 系数比较大，从而量化后变换为 0 的概率比 AC 系数要小。量化操作其实非常简单，就是除法操作。计算公式如下：



极客时间

在量化过程中，最重要的就是 QStep（用户一般接触到的是 QP，两者可以查表转换）。

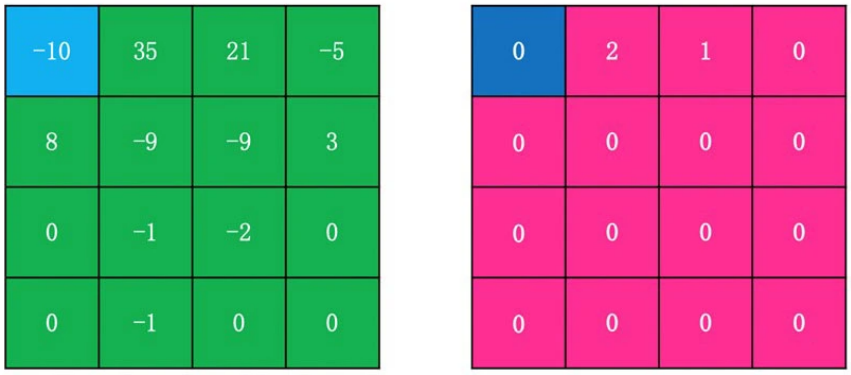
其中，在 H264 中 QP 和 QStep 之间的转换表格如下：

QP	QStep	QP	QStep	QP	QStep	QP	QStep
0	0.625	14	3.25	28	16	42	80
1	0.6875	15	3.5	29	18	43	88
2	0.8125	16	4	30	20	44	104
3	0.875	17	4.5	31	22	45	112
4	1	18	5	32	26	46	128
5	1.125	19	5.5	33	28	47	144
6	1.25	20	6.5	34	32	48	160
7	1.375	21	7	35	36	49	176
8	1.625	22	8	36	40	50	208
9	1.75	23	9	37	44	51	224
10	2	24	10	38	52		
11	2.25	25	11	39	56		
12	2.5	26	13	40	64		
13	2.75	27	14	41	72		

极客时间

通常 QStep 值越大，DC 系数和 AC 系数被量化成 0 的概率也就越大，从而压缩程度就越大，但是丢失的信息也就越多。这个值太大了会造成视频出现一个个块状的效应，且严重的时候看起来像马赛克一样；这个值比较小的话，压缩程度也会比较小，从而图像失真就会比较小，但是压缩之后的码流大小就会比较大。

我们通过一个例子来看一下量化的结果。



DCT变换后的块　　QP取30（QStep为20）量化后（使用H264标准）

极客时间

这就是常规的变换和量化的计算过程。实际上 H264 里面的变换和量化是这样的吗？原理上是的，但是实际计算过程变了。因为 DCT 变换过程中涉及到浮点运算，在不同机器上解码会因为精度问题产生漂移导致误差。同样，量化过程有除法运算，大多数时候其结果还是浮点型的数字，在不同机器上解码也会有误差。

H264 中的 DCT 变换和量化

H264 为了减少这种浮点型运算漂移带来的误差，将 DCT 变换改成了整数变换，DCT 变换中的浮点运算和量化过程合并，这样就只有一次浮点运算过程，以此来减少不同机器上浮点运算产生的误差。下面我们来看看 H264 中的变换和量化。

H264 的整数变换和量化

我们知道常规的 DCT 变换的矩阵计算方式如下：

$$Y = AXA^T$$

其中:
Y是变换后的4x4变换块
X是4x4残差块

$$Y = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) \end{bmatrix} X \begin{bmatrix} \frac{1}{2} & \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \frac{1}{2} & \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) \\ \frac{1}{2} & \sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\frac{1}{2} & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) & -\frac{1}{2} & \sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) \\ \frac{1}{2} & -\sqrt{\frac{1}{2}}cos(\frac{\pi}{8}) & \frac{1}{2} & -\sqrt{\frac{1}{2}}cos(\frac{3\pi}{8}) \end{bmatrix}$$

极客时间

而在 H264 中，我们通过下面的推导过程，将 DCT 变换一步步修改为整数变换。最后 H264 中的 DCT 变换就变成了整数变换。其矩阵的计算方式如下：

$$Y = AXA^T$$

变换为：

$$Y = C_f X {C_f}^T \otimes E_f$$

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} X \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

其中：$a = \frac{1}{2}$ $b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8})$ $\otimes$ 表示点乘

极客时间

我们将点乘左边的部分取出来，就是 H264 中的整数变换了。公式如下：

整数变换：

$$Y' = C_f X {C_f}^T$$

$$Y' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} X \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}$$

极客时间

我们同样使用上面 DCT 变换的例子来做一下整数变换。其结果如下：

9	2	−5	−5
14	−1	−11	−7
15	−3	−18	−9
16	−3	−22	−11

4x4残差块

−39	231	83	−16
52	−27	−54	28
1	−7	−9	−6
1	−16	−7	−3

整数变换后的块

极客时间

在前面整数变换里，DCT 变换中的点乘部分被拿出来了，这一部分的计算被合并到了 H264 的量化过程中。因此 H264 的量化过程如下所示：

$$Z = round(\frac{y}{QStep})$$

变成：

$$Z = round(\frac{y'}{QStep}PF)$$

其中：

$$PF = \begin{cases} a^2 & (0,0),(2,0),(0,2),(2,2) \\ \frac{b^2}{4} & (1,1),(1,2),(3,1),(3,3) \\ \frac{ab}{2} & \text{其余位置} \end{cases}$$

$a = \frac{1}{2}\ b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8})$

我们再将PF和QStep合并成一个MF。可以得到：

$$Z = round(\frac{y'}{2^{15+floor(QP/6)}}MF)$$

$$MF = \frac{PF}{QStep}2^{15+floor(QP/6)}$$

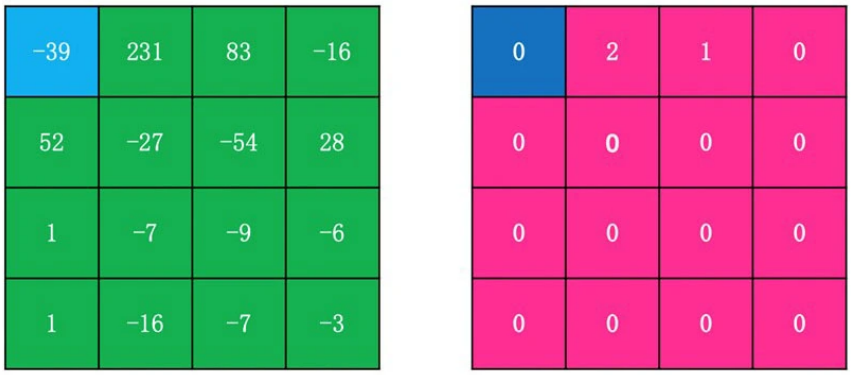
极客时间

其中，MF 我们一般都是通过表格查询得到。表格如下。其中，对于 QP 大于 5 的情况，使用 QP = QP % 6 进行查询。

QP	(0,0),(2,0), (0,2),(2,2)位置	(1,1),(1,3), (3,1),(3,3)位置	其余 位置
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

极客时间

同样，上面整数变换之后，我们用 H264 的量化公式对其进行量化后得到的结果如下：



整数变换后的块　　QP取30量化后（使用H264标准）

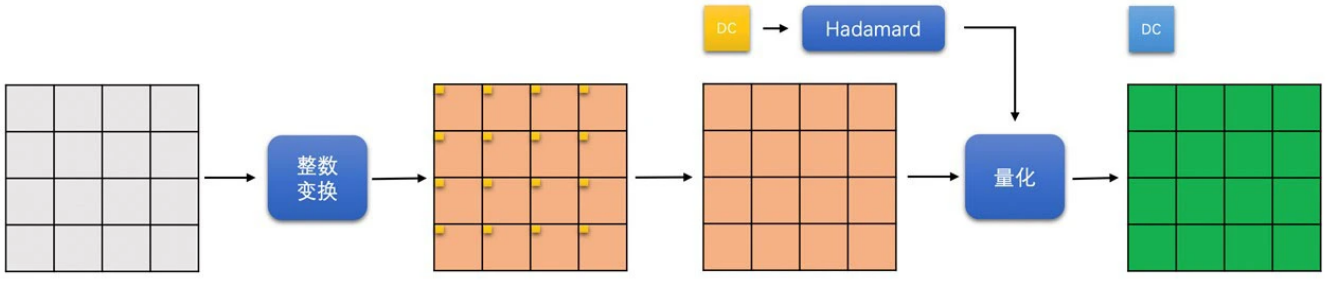
极客时间

我们可以看到虽然 H264 的 DCT 变换和量化过程跟常规的 DCT 变换和量化不一样，但是最后量化的结果其实还是一样的。这也是符合预期的。毕竟它们的“目的地”还是一样的，只是“走的路”稍微有些不同而已。

好了，这就是 H264 中的 DCT 变换和量化的基本原理。接下来我们来看看 H264 各模式块的 DCT 变换和量化过程具体是怎么样的。

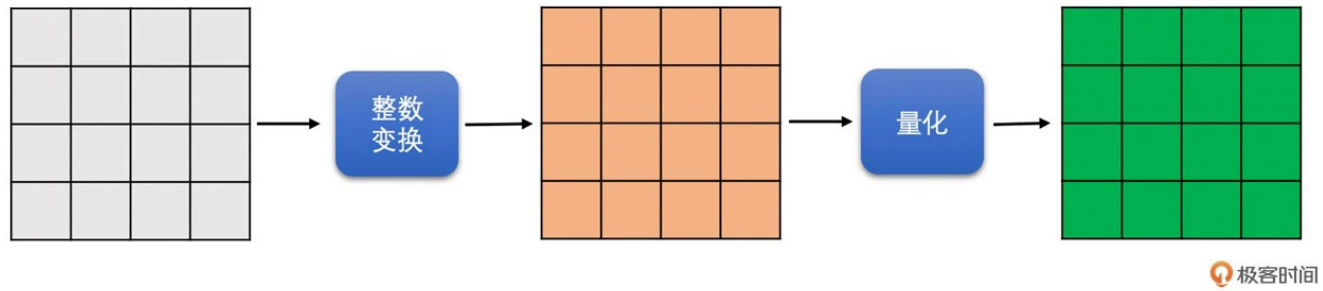
H264 各模式块的 DCT 变换和量化过程

1. 亮度 16x16 帧内预测块
   亮度 16x16 块，首先被划分成 16 个 4x4 的小块做整数变换。变换之后将 16 个 4x4 小块的 DC 系数都拿出来，组成一个 4x4 的 DC 块，再对这个 4x4 的 DC 块进行 Hadamard 变换。然后，再总体进行量化操作。



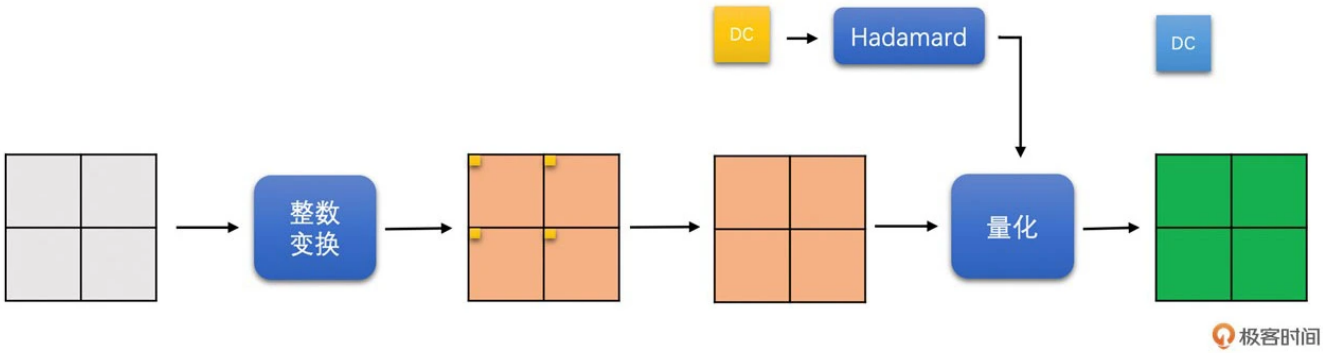
极客时间

2. 其它模式亮度块
   对于除亮度 16x16 帧内预测块之外的其它亮度块，都是直接划分成 4x4 的块进行整数变换，之后再进行量化操作就可以了。



3. 色度块

对于 YUV420 图像，色度块大小是 8x8。我们先将 8x8 色度块划分成 4 个 4x4 的小块做整数变换。变换之后将 4 个小块的 DC 系数拿出来，组成一个 2x2 的 DC 块，再对这个 2x2 的 DC 块进行 Hadamard 变换。最后总体进行量化操作。



小结

我们今天一开始主要讲解了 DCT 变换的基本原理。DCT 变换主要是将图像从空域转换到频域，并将图像的高频和低频信息分离开来。虽然高频信息数据多，但是幅值比较小。这样高频信息在量化的过程中能够比较容易被减少。这样可以比较有效地减少图像的视觉冗余，从而达到压缩的目的。

接着，我们简单地介绍了一下量化的原理。量化其实就是一个除法操作。通过除法操作就可以将幅值变小，而高频信息幅值比较小，就比较容易被量化成 0，这样就能够达到压缩的目的。

在讲变换的原理的时候，我们还讲到了一个前面提到了好几次的 Hadamard 变换。Hadamard 变换在 H264 的 16x16 帧内亮度块和 8x8 色度块中会被用到。但是 Hadamard 在率失真优化做模式选择的时候使用的更多。基本上各种视频编码都或多或少会用到它来做率失真优化。

在 H264 标准中，我们不会直接使用标准的 DCT 变换和量化。为了减少多次浮点型运算在解码端产生漂移的问题，H264 使用整数变换代替 DCT 变换。DCT 变换中的浮点运算部分跟量化过程进行合并，将两次浮点型运算变成一次，从而减少误差。

在最后，我们简单介绍了 H264 标准中不同模式亮度块和色度块的 DCT 变换和量化的过程。其中需要注意的就是亮度 16x16 帧内预测块和色度 8x8 的 DC 系数会单独拿出来组成一个新的 DC 块，我们会先对这个 DC 块进行 Hadamard 变换之后再做量化操作。

思考题

为什么我们在率失真优化的过程中会用 Hadamard 变换之后的块做大小预估？

你可以把你的答案和疑惑写下来，分享到留言区，与我一起讨论。下节课再见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 帧间预测：如何减少时间冗余？

下一篇 09 | RTP & RTCP：如何正确地将视频装进RTP中？

精选留言 (3)

写留言

晓龙
2021-12-09

我有点理解为什么要对DC块在做一次hadamard变换，因为DC块的值比较大，在做一次hadamard变换，可以进一步将低频与高频分离，减少视觉冗余，老师，是这样理解么？

晓龙

2021-12-09

亮度块和色度块中的DC块比AC块多了一次hadamard变换，为什么？第一步整数变换，不是已经将所有的块从空域变换到频域了么？这里对DC块在做一次hadamard变换，相当于对已经转换到频域的块，在做一次频域转换？这里不太理解，请老师解答下

展开

ForwardsHao

2021-12-08

“亮度 16x16 帧内预测块和色度 8x8 的 DC 系数会单独拿出来组成一个新的 DC 块，我们会先对这个 DC 块进行 Hadamard 变换之后再做量化操作” 这个收益是什么呢？对DC信息是存了两份吗？还是做完Hadamard变换后还会把DC系数放回预测块内？如果是放回的话，那我推测收益是对DC系数也可以做很大程度的压缩，而后可以通过反Hadamard变换还原出不至于影响太多低频信息

展开